Multigrid preconditioning for a space-time spectral-element discontinuous-Galerkin solver

Matteo Franciolini[*] and Scott Murman[†]

NASA Ames Research Center, Moffett Field, 94035 (CA), United States

In this work we examine a multigrid preconditioning approach in the context of a high-order tensor-product discontinuous-Galerkin spectral-element solver. We couple multigrid ideas together with memory lean and efficient tensor-product preconditioned matrix-free smoothers. Block ILU(0)-preconditioned GMRES smoothers are employed on the coarsest spaces. The performance is evaluated on nonlinear problems arising from unsteady scale-resolving solutions of the Navier–Stokes equations: separated low-Mach unsteady flow over an airfoil from laminar to turbulent flow. A reduction in the number of fine space iterations is observed, which proves the efficiency of the approach in terms of preconditioning the linear systems, however this gain was not reflected in the CPU time. Finally, the preconditioner is successfully applied to problems characterized by stiff source terms such as the set of RANS equations, where the simple tensor product preconditioner fails. Theoretical justification about the findings is reported and future work is outlined.

I. Introduction

In recent years the increasing availability of High Performance Computing (HPC) resources strongly promoted the widespread evaluation of Large Eddy Simulation (LES) turbulence modelling approaches. In particular, LES based on space-time discontinuous-Galerkin (dG) discretizations show very promising results on a variety of cases,[1–4] thanks to the arbitrary high order of accuracy of the method in space and time, the use of entropy-stable schemes and the ease of implementation of adjoint sensitivities. Space-time solvers are typically thought too expensive for practical engineering simulations since they require to solve a fully coupled nonlinear system per each time step of the discretization. The number of non-zeros of a matrix scales as $\mathcal{O}(N^{2d})$, where N is the spatial order of accuracy and d is the number of dimensions of the problem. Thus, in 3D the memory required, as well as the computational cost for the matrix assembly scales as $\mathcal{O}(N_t^2 N^6)$, where N_t is the number of basis functions in the time direction. This makes the use of high order space-time methods prohibitively expensive.

Previous work by the authors[5] focused on the implementation of efficient solution strategies to overcome the limitations of memory and computational costs associated with space-time discretizations. The solver circumvents the memory limitations by using a matrix-free Newton-Krylov iterative solver, where the linearization of the residual along the direction of the solution update is computed directly. In addition, tensor-product preconditioner based on a diagonalized Alternating-Direction-Implicit (ADI) scheme maintains an optimal computational efficiency. This preconditioning approach does not require the allocation of a residual Jacobian block, maintaining the memory footprint equal to that of an explicit solver. However, the possibility of maintaining a separable tensor-product factorization of the matrix cannot be easily generalized to elliptic operators or source terms of arbitrary form. Recent approaches aimed to extend those limitations appeared in the recent literature,[6, 7] where the best approximate tensor-product form of the inverse of the elemental block Jacobian matrix is solved through a matrix-free singular value decomposition algorithm (SVD) or an optimization problem. However, the application of those techniques to general problems remains a subject of active research.

Nevertheless, the preconditioning strategy remains local-to-each-element and thus it is unable to precondition stiff systems of equations, for example those arising from low-Mach number simulations, where

[*]USRA Fellow, NASA Advanced Supercomputing Division, matteo.franciolini@nasa.gov
[†]NASA Advanced Supercomputing Division, scott.murman@nasa.gov

pressure perturbations travel much faster than velocity perturbations and thus a tighter coupling of the elements exists. In fact, recalling that the block-Jacobi (BJ) preconditioner involves the element-wise LU factorization of the on-diagonal of the residual Jacobian block, such tensor-product preconditioners can be seen as an approximate factorization of the BJ preconditioner when tensor-product basis functions are considered.

This work examines whether the multigrid approach provides a benefit to the above tensor-product preconditioner for solving the Navier-Stokes equations. Multigrid methods are powerful methods for elliptic and parabolic problems. Such methods were first proposed in a dG context as a nonlinear solver by Helenbrook *et al.*,[8] Bassi and Rebay,[9] Fidkowski *et al.*[10] The possibility of using multigrid operators as a preconditioner of an iterative solver was also explored in the context of steady compressible flows.[11,12] In these works, the algorithm is reported as the most efficient and scalable if compared to single-grid preconditioners, and a large reduction in the number of iteration to reach convergence was achieved. More recently, an h-, p- and h/p-multigrid preconditioners were proposed[13–15] in the context of steady and unsteady incompressible flows. With the idea of targeting simulations at a very high order of accuracy, we here propose to implement the p-version of the multigrid preconditioner in a space-time spectral-element solver. Previous works showed optimal scalability, low memory footprint and applicability to complex flows and configuration.[14,16] We here aim at extending the approach to arbitrary orders of accuracy in a space-time spectral element solver by the use of tensor-product preconditioning strategies on the fine-space smoothers. On the coarsest space, block-based preconditioning methods are employed for the iterative smoothers in order to ensure an adequate resolution of the coarse space problem. Inter-element and inter-partition couplings, which are missing when using tensor-product preconditioning approaches, are ensured by the use of Block ILU(0), which is inexpensive to compute on the coarsest space.

The paper includes an extensive validation of the numerical strategy on test cases involving high-order accurate solutions on a set of stiff computational settings involving low-Mach number flows, from laminar low-Reynolds to turbulent high-Reynolds regimes on stretched and skewed mesh elements. The effects of the hyperparameters such as the number of levels of the multigrid iteration, the number of smoothing iterations in each levels, as well as preconditoning type for each level are be explored on the computational efficiency of the method. Strong reduction in number of fine-space iterations is observed for diffusion dominated regimes. Finally, the preconditioner is successfully applied to the solution of the RANS equations, which proves the efficacy of the approach where the single-grid tensor product preconditioner fails. A theoretical justification about the findings is reported, which considers implementation details and future directions.

II. Background: the numerical framework

In this section the space and time discretizations of the Navier–Stokes equations are briefly introduced together with a detailed description of the main building blocks of the p-multigrid preconditioner.

A. The space-time dG discretization

Space discretization is based on the discontinuous Galerkin method described by Diosady and Murman.[3] We here summarize the main features. The method employs an entropy stable formulation of the compressible Navier–Stokes equations. Entropy-stability is obtained through the change of variables $\boldsymbol{u} = \boldsymbol{u}(\mathbf{v})$, where $\boldsymbol{u}$ are the conservative and $\mathbf{v}$ are the entropy variables defined as follows:

$$\mathbf{v} = \begin{bmatrix} -\frac{s}{\gamma-1} + \frac{\gamma+1}{\gamma-1} + \frac{\rho E}{p} \\ \frac{\rho \boldsymbol{V}}{p} \\ -\frac{\rho}{p} \end{bmatrix} \qquad \boldsymbol{u} = \begin{bmatrix} \rho \\ \rho \boldsymbol{V} \\ \rho E \end{bmatrix}, \tag{1}$$

we rewrite the Navier-Stokes equations as:

$$\boldsymbol{A}_0 \mathbf{v}_{,t} + \bar{\boldsymbol{A}} \nabla \mathbf{v} - \nabla \cdot (\bar{\bar{\boldsymbol{K}}} \nabla \mathbf{v}) \quad = \quad 0 \tag{2}$$

with symmetric $\boldsymbol{A}_0 = \boldsymbol{u}_{,\mathbf{v}}$, $\bar{\boldsymbol{A}} = \boldsymbol{f}^I_{,\boldsymbol{u}} \boldsymbol{A}_0 = \boldsymbol{f}^I_{,\mathbf{v}}$ and $\bar{\bar{\boldsymbol{K}}} = \boldsymbol{f}^V_{,\nabla \boldsymbol{u}} \boldsymbol{A}_0 = \boldsymbol{f}^V_{,\nabla \mathbf{v}}$.[17] Note that by $\boldsymbol{f}^I, \boldsymbol{f}^V$ we denote the inviscid and viscous fluxes, respectively.

We discretize (2) as follows. The domain Ω is partitioned into possibly curved non-overlapping elements, K, while the time is partitioned into intervals, here denoted as time-slabs, defined as $I^n = [t^n, t^{n+1}]$. The polynomial approximation space is defined as $\mathcal{V}_\kappa = \{\boldsymbol{w}, \boldsymbol{w}|_{K\times I} \in [\mathcal{P}_\kappa(K\times I)]^m\}$, with κ equal to the order of polynomial approximation in the space-time domain consisting of piece-wise polynomial functions, where $m = 5$ is the number of equations. We seek a solution $\mathbf{v} \in \mathcal{V}_h$ such that the weak form

$$\begin{aligned} \boldsymbol{r}(\boldsymbol{w},\mathbf{v}) \;\; = \;\; & \sum_\kappa \Big\{ \int_I \int_\kappa - \left(\boldsymbol{w}_{,t}\cdot\boldsymbol{u} + \nabla\boldsymbol{w}\cdot(\boldsymbol{f}^I - \boldsymbol{f}^V)\right) \\ & + \int_I \int_{\partial\kappa} \boldsymbol{w}\cdot(\widehat{\boldsymbol{f}^I\cdot\boldsymbol{n}} - \widehat{\boldsymbol{f}^V\cdot\boldsymbol{n}}) \\ & + \int_\kappa \boldsymbol{w}(t_-^{n+1})\cdot\boldsymbol{u}(t_-^{n+1}) - \boldsymbol{w}(t_+^n)\cdot\boldsymbol{u}(t_-^n) \Big\} = 0 \end{aligned} \tag{3}$$

is satisfied for all $\boldsymbol{w} \in \mathcal{V}_h$, where $\boldsymbol{u} = \boldsymbol{u}(\mathbf{v})$ as given above. Here $\widehat{\boldsymbol{f}^I\cdot\boldsymbol{n}}$ and $\widehat{\boldsymbol{f}^V\cdot\boldsymbol{n}}$ denote numerical flux functions approximating the inviscid and viscous fluxes, respectively, while $\boldsymbol{n}$ is the outward pointing normal vector. In this work, the inviscid flux is discretized using the method of Ismail and Roe,[18] while the viscous flux is discretized using an interior penalty method with stabilization parameters computed following the second form of the Bassi and Rebay scheme. We remark that tensor-product basis functions[19] are employed to reduce the computational complexity of the residual evaluation through the use of the sum-factorization approach. The computational costs can be further reduced by the use of optimized numerical kernels to maximise the throughput of the application on modern CPU architectures.

B. Matrix-free implicit time integration

The space-time fully coupled linear system is solved using a matrix-free Newton-Krylov algorithm. A single Newton update is obtained through the solution of

$$\left(\frac{\partial \boldsymbol{R}^*}{\partial \boldsymbol{U}}\right)\Delta\boldsymbol{U}^k \;\; = \;\; -\boldsymbol{R}^*(\boldsymbol{U}^k), \tag{4}$$

with $\boldsymbol{R}^*(\boldsymbol{U})$ is the unsteady residual and $\Delta\boldsymbol{U}^k$ is the Newton's update. Eq. (4) represents a linear system that we solve using a preconditioned restart Generalized Minimal Residual (GMRES) algorithm. One advantage of such a method is that the residual Jacobian matrix $\partial\boldsymbol{R}^*/\partial\boldsymbol{U}$ is not required explicitly, only action on the residual update. This task is accomplished within the following steps:

1. Evaluation of the state, gradient, linearized state and linearized gradient ($\boldsymbol{U}$, $\nabla\boldsymbol{U}$, $\boldsymbol{V}$ and $\nabla\boldsymbol{V}$) at the quadrature point.
2. Evaluation of the linearized fluxes ($\boldsymbol{F}_{\text{Lin}} = \frac{\partial\boldsymbol{F}}{\partial\boldsymbol{U}}\boldsymbol{V} + \frac{\partial\boldsymbol{F}}{\partial\nabla\boldsymbol{U}}\nabla\boldsymbol{V}$) at the quadrature points.
3. Multiplication of the linearized fluxes with the gradient of the basis functions.

This approach is more expensive than the finite-difference approximation of Knoll and Keyes.[20] However, the additional cost eliminates the dependence on a arbitrary small numerical perturbation ϵ and it is consistent with the solution of an adjoint (dual) problem. In addition, when computing the exact Fréchet derivative, we are able to offset this additional cost by exploiting the optimized numerical kernels based on the sum-factorization approach for the steps 1 and 3. It is also worth noticing that the approach does not pollute the GMRES nonlinear residual when small tolerances on the solution are required.[21,22] The use of a matrix-free implementation decouples the use of the Jacobian to compute the Frechet derivatives with respect to its use for preconditioning, and increases the flexibility of the solver.

III. Multigrid preconditioning

The idea is to use a multigrid preconditioning strategy coupled with a Flexible Generalized Minimal Residual Method (FGMRES). Multigrid allows the use of iterative solvers to smooth-out the high-frequency component of the error with respect to the unknown exact solution. As typical iterative solvers are not effective at damping low-frequency error components, the iterative solution of coarser problems is exploited to circumvent this issue, shifting the low-frequency modes towards the upper side of the spectrum. This simple and effective strategy provides satisfactory rates of convergence all along the iterative process.

As the work aims at obtaining solutions with high-order polynomials on large and possibly curved mesh elements, and targets the use of the solver on large HPC facilities where very few elements per partition are employed, a solution to build coarse spaces effectively involves coarse order discretizations κ_ℓ of the problem of order κ, known in the literature as p-multigrid method. The strategy shows some advantages over h-multigrid approaches on agglomerated mesh elements beyond the ease of implementation, as the intergrid transfer operators and the matrix assembly routines are applied in a local-to-each-element fashion, and thus are ideally scalable. We consider L coarse levels spanned by the index $\ell = 1, ..., L$ and indicate the fine and coarse levels with $\ell = 0$ and $\ell = L$, respectively. The polynomial degree of level ℓ is κ_ℓ and the polynomial degrees of the coarse levels are chosen such that $\kappa_\ell < \kappa_{\ell-1}$, $\ell = 1, ..., L$, with $\kappa_0 = \kappa$. Accordingly the polynomial spaces associated to the coarse levels read $\mathcal{V}_\kappa$. The coarse problems corresponding to (4) are in the form

$$\left(\frac{\partial \boldsymbol{R}^*}{\partial \boldsymbol{U}}\right)_\ell \Delta \boldsymbol{U}_\ell \quad = \quad -\widetilde{\boldsymbol{R}}^*_\ell \tag{5}$$

where the subscript ℓ denotes quantities evaluated on the coarse level of the multigrid cycle, $\Delta \mathbf{U}_\ell, \widetilde{\boldsymbol{R}}^*_\ell \in \mathcal{V}_{\kappa_\ell}$ are the unknown function and the known right-hand side obtained through restriction, respectively.

A crucial aspect for the efficiency of the p-multigrid iteration is related to the computational cost of building coarse grid operators. Two main approaches have been widely considered in the literature. The first, typically employed within solvers relying on matrix-based methods, involves the construction of the matrix operators on the fine space and their restriction to coarse spaces through a Galerkin projection. This implementation avoids the recomputation of the Jacobian matrix on coarse spaces reducing the matrix assembly time, and it is typically referred in the literature as *inherited* method. On the other hand, coarse spaces can be obtained by rediscretization of the fine space problem on a lower order polynomial basis, also known as *non-inherited* method. As demonstrated by Antonietti *et al.*,[23] the latter approach allows for better multigrid convergence rates, at the expenses of higher computational costs due to the rediscretization of the coarse space operators. However, the second method is particularly easy to apply within a fully matrix-free framework, where it can be implemented just by computing the residual's linearization at polynomial order κ_ℓ around the L2-projection of the fine space state vector. The application of such linearization is done following the steps described in Section II.B. Prolongation and restriction operators are defined according to Franciolini *et al.*[24]

IV. The p-multigrid iteration

In this section we provide an overlook of the sequence of operations involved in p-multigrid iterations. The recursive p-multigrid $\mathcal{V}$-cycle for the problem of Eq. (5) on level ℓ are reported in Algorithm 1. To obtain an application of the p-multigrid preconditioner the multilevel iteration is invoked on the fine problem, Eq. (4). One MG$_\mathcal{V}$ iteration requires two applications of the smoother on the finest level (one pre- and one post-smoothing) and one application of the coarse level smoother independently from the number of levels.

In this work the p-multigrid $\mathcal{V}$ cycle iteration is applied for the numerical solution of the linearized equations systems arising in Newtons method. In the context of such problems, we seek for the best performance employing p-multigrid and tuning preconditioners and smoothing options. In this work, all the smoothers of the multigrid strategy are GMRES, and thus two nested Krylov iterative solvers are invoked to maximise the stability of the smoothers. In this setting, the outer solver follows the flexible GMRES implementation,[25] since the action of the multigrid linear solver needs to be stored at each iteration. The matrix-free implementation is employed to maintain a low memory footprint and high arithmetic intensity.

We remark that only if cheap preconditioners such as tensor-product ADI or Block Jacobi are coupled with a matrix-free GMRES smoother on the finest space, a limited memory footprint of the solver can be achieved. As demonstrated in previous work by the authors,[14] such a strategy can be conveniently used for practical simulations without spoiling the convergence rates of the multigrid iteration. Here we exclusively test the Block-ILU(0) preconditioner (BILU) for the coarse space smoother to ensure a satisfactory convergence rate. We point out that, since the number of non-zeros of the primal Jacobian matrix scales as k^{2d}, the memory footprint of a coarse space matrix can be less than 1% than the memory allocated for the fine space, and thus the overall memory footprint remains limited.

Algorithm 1 $\overline{\mathbf{U}}_\ell = \mathrm{MG}_\mathcal{V}(l, \widetilde{\boldsymbol{R}}^*_\ell, \mathbf{U}_\ell)$

```
if $(\ell = L)$ then
    $\overline{\mathbf{U}}_\ell = \mathrm{solve}(\mathrm{lin}_\ell, \widetilde{\boldsymbol{R}}^*_\ell, 0)$
end if
if $(\ell < L)$ then
    Pre-smoothing:
    $\overline{\mathbf{U}}_\ell = \mathrm{smooth}(\mathrm{lin}_\ell, \widetilde{\boldsymbol{R}}^*_\ell, \mathbf{U}_\ell)$

    Coarse grid correction:
    $\mathbf{D}_\ell = \widetilde{\boldsymbol{R}}^*_\ell - \mathrm{lin}_\ell(\overline{\mathbf{U}}_\ell)$
    $\mathbf{D}_{\ell+1} = \mathcal{I}_\ell^{\ell+1}\mathbf{D}_\ell$
    $\boldsymbol{E}_{\ell+1} = \mathrm{MG}_\mathcal{V}(\ell+1, \mathbf{D}_{\ell+1}, 0)$
    $\widehat{\mathbf{U}}_\ell = \overline{\mathbf{U}}_\ell + \mathcal{I}_{\ell+1}^{\ell}\boldsymbol{E}_{\ell+1}$

    Post-smoothing:
    $\overline{\mathbf{U}}_\ell = \mathrm{smooth}(\mathrm{lin}_\ell, \widetilde{\boldsymbol{R}}^*_\ell, \widehat{\mathbf{U}}_\ell)$
end if
return $\overline{\mathbf{U}}_\ell$
```

V. Numerical Results

The current work uses the NACA 4412 airfoil section near maximum lift in order to assess the performance of the multigrid approach. This roughly corresponds to the experimental data by Wadcock,[26] however the goal here is not to validate against a particular dataset[a], but rather provide general trend information for a relevant flow physics. The flow conditions are $Ma = 0.1$, $\alpha = 12°$, with the computed Reynolds numbers of $Re = 500, 5000, 50\text{k}$, and $Re = \infty$. In addition to the variation with Reynolds number, the performance trends with increasing Courant-Fredrichs-Lewy (CFL) number ($CFL = c\frac{\Delta t N}{\Delta N_t}$) are examined.

The numerical tests use roughly 73k, 368k, 2.1M dof per equation at $Re = 500, 5000, 50\text{k}$, respectively, with a three-dimensional (planar) configuration. Contours of velocity magnitude in the center of the planar section are presented in Fig. 1 at the relevant Reynolds number. The pressure side of the airfoil remains attached at all Reynolds numbers. At $Re = 500$ there is a laminar boundary layer separation on the airfoil suction side, and the wake remains laminar. At $Re = 5000$, there is again a laminar boundary layer separation, with transition to turbulence in the free shear layer. At $Re = 50\text{k}$ the the suction-side boundary layer transitions before a turbulent separation near the trailing edge.

Performance of the spectral multigrid approach is measured relative to the single grid approach on the same problem with the same numerical settings for convergence tolerance, timestep, *etc.* Currently, only two levels of multigrid are tested, $N = 4$ for the fine space and $N = 2$ for the coarse space. Different combinations of pre- and post-smoothing iteration counts are tested, and the best performance selected. Here,

[a]Validation data for the *eddy* solver for the NACA 4412 is available in Carton de Wiart *et al.*[2]

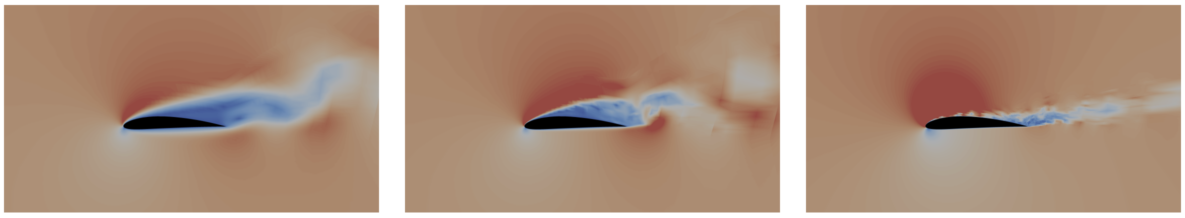
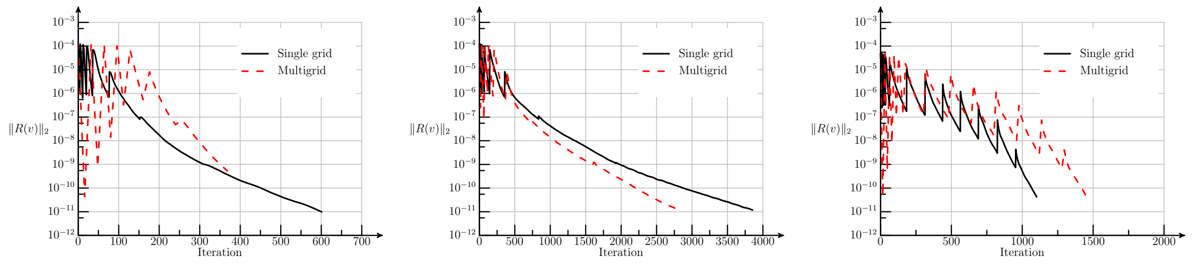

(a) $Re_c = 500$ (b) $Re_c = 5000$ (c) $Re_c = 50\text{k}$

Figure 1. Instantaneous snapshot of velocity magnitude contours for the NACA 4412 airfoil test case.

the coarse space is always converged to machine precision. The intent here is to evaluate the performance potential of spectral multigrid in the current tensor-product formulation, not prescribe optimal procedures. A practical implementation would require some automated mechanics to choose the best combination of pre- and post-smoothing iterations counts, as well as an appropriate coarse grid solution tolerance, to minimize the time-to-solution.

Figure 2 presents the full nonlinear convergence behavior of the Newton-Krylov solver for a single time slab over a sampling of conditions. In terms of evaluating the performance of the multigrid preconditioner, a couple of issues are apparent. First, the initial phase of the Newton-Krylov solve is dominated by highly nonlinear behavior. The single-grid solver has an automated method for controlling this behavior, which is disabled in the current runs in an attempt to equalize the results.[b] Even with this adjustment, the nonlinear startup region is highly variable between runs, with this extended to the entire convergence history as the Reynolds number increases and the system becomes stiffer. This makes it difficult to get consistent comparisons between fine grid iteration count and/or CPU time between the single and multigrid methods. As the goal in this work is to evaluate the preconditioning for the linear solver, not evaluate the nonlinear



(a) $Re_c = 5000$, Block-Jacobi (b) $Re_c = 5000$, ADI (c) $Re_c = 50\text{k}$, Block-Jacobi

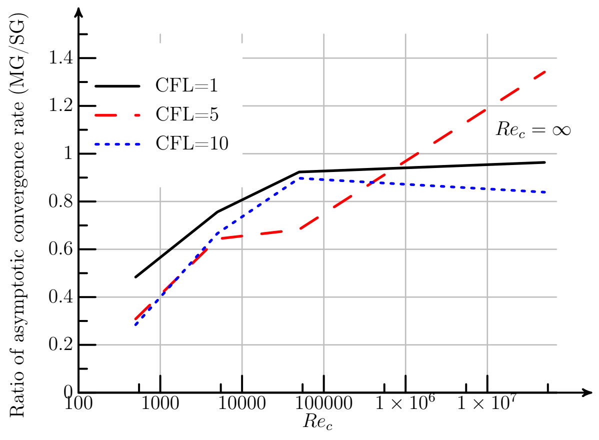
Figure 2. Convergence of the nonlinear Newton-Krylov solver over a single space-time step.

solver, we use the asymptotic convergence rate of the Newton-Krylov solve to compare the schemes. This is linearly related to the number of fine grid iterations, and can be consistently and automatically calculated from the data. All of the trends discussed herein were a-posteriori confirmed by solving the linear advection-diffusion equation.

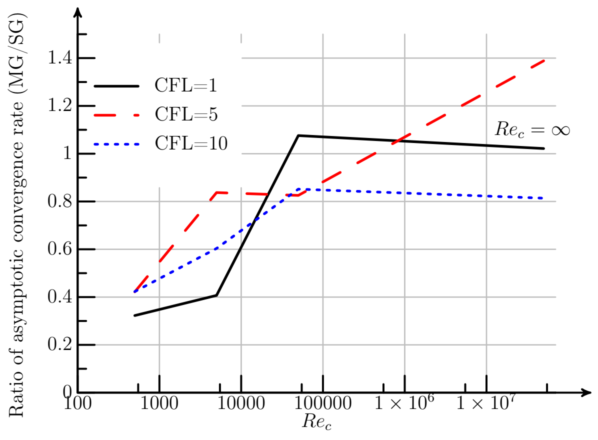
Figure 3 presents results using the BJ preconditioner on the fine space, paired with BILU on the coarse space, and also the ADI preconditioner on the fine space paired with BILU on the coarse space[c]. The results indicate that multigrid does increase the convergence rate (reduce the number of fine grid iterations) in the lower Reynolds number cases, where the flow is more diffusion dominated, and that the benefits drop off with increasing Reynolds numbers, as expected. There is little correlation in convergence rate due to changing CFL over the range examined.

The reduced fine grid iterations does not lead to a commensurate reduction in overall CPU time, even accounting for the nonlinear behavior. Across the Reynolds number and CFL range the multigrid CPU time is roughly identical to the single grid results. The previous dG spectral multigrid implementations discussed in the introduction demonstrated more consistent performance than the current results. Namely, the reduction in fine grid iterations directly correlated with a reduction in overall CPU time. The reason for this is the efficiency of the relative schemes. Previous studies use a general Galerkin implementation which utilizes a element-wise block diagonal matrix for the fine space smoother, rather than a matrix-free tensor-product implementation. The generic scaling of these two approaches with increasing order of accuracy is presented in Fig. 4. The tensor-product implementation used in the current work scales much better with increasing order, at a cost of reduced efficiency at low order. Thus, the cost of the fine grid residual is cheaper for the tensor-product formulation, while the coarse grid is (relatively) more expensive than the more general approach. This is exactly the opposite to the desired behavior to exploit multigrid savings, and leads to the observed trends for overall CPU time.

[b]There is currently insufficient experience with the mutigrid scheme to develop a similar controller.
[c]$Re_c = \infty$ is plotted at $Re_c = 5\text{x}10^7$



(a) Block-Jacobi



(b) ADI

Figure 3. Relative asymptotic convergence rate over a single time step for the NACA 4412 airfoil.

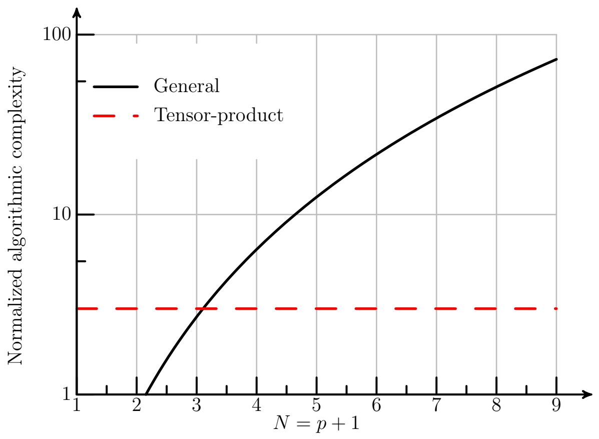


Figure 4. Algorithmic complexity scaling for spectral-element implementations.

VI. Application to the RANS equations

The preconditioning strategy is applied to the solution of the set of RANS equations. The Spalart-Allmaras turbulence model we considered is the one proposed by Allmaras *et al.*,[27] suitable to be implemented in a high-order solver. The test is the flow around a NACA 0012 airfoil at high angle of attack ($Ma = 0.1$, $Re_c = 1.3M$ and $\alpha = 60°$). The solution is first initialised using an N=2 simulation, then projected to an N=4 state, and then converged using an unsteady process until reaching a steady state. Figure 5 presents the convergence history over a single time step using the ADI preconditioner in a single grid simulation, and then the ADI preconditioner on the fine space ($N = 4$) and BILU on the coarse space ($N = 2$). As the ADI tensor-product preconditioner cannot handle general source terms, the convergence of the single grid simulation is very poor. In contrast, the multigrid approach, which utilizes the full Jacobian BILU on the coarse space is able to efficiently converge to machine epsilon.

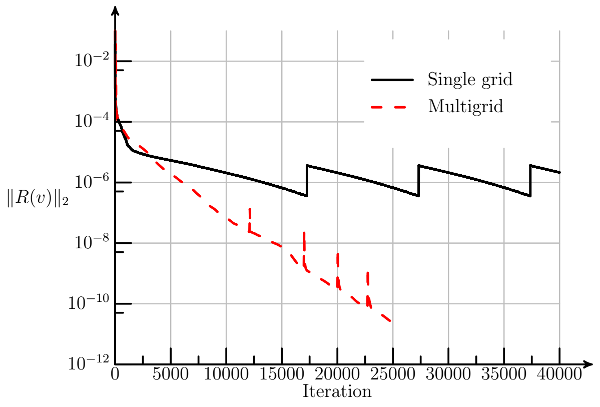


Figure 5. Newton-Krylov convergence history over a time slab for the NACA 0012 RANS simulation

VII. Future Work

The paper presents a p-multigrid preconditioner strategy applied to the solution of linear system arising from space-time discontinuous-Galerkin discretizations. The algorithm is based on a matrix-free implementation of both the outer FGMRES solver as well as of the smoothers. Element-wise Block Jacobi preconditioned smoothers, as well as tensor-product preconditioning strategies have been employed at the highest polynomial orders to maximise the computational efficiency.

The results show promising improvements on the lower-Reynolds cases with increases in convergence rate of up to 4x. This gain reduces by increasing the Reynolds number, suggesting that the coarse correction is most useful on diffusion dominated areas of the flow. On the other hand, the CPU time was found not to reflect the same improvements, primarily due to the overhead costs of the tensor-product formulation when applied at low order of accuracy.

Future work will be focused on reducing the costs of the coarse grid correction by implementing a general element solver to apply the residual's linearization at lower computational cost. The use of sparse and distributed matrix-vector product will also be investigated, thus utilizing a proper matrix-based method on the coarsest level of the multigrid cycle. Finally, the utility of the multigrid preconditioner will be assessed on general, mixed element meshes where the ADI tensor-product preconditioner does not perform as well as the current hexahedral mesh elements.[19]

While the current work focuses on the Navier-Stokes equations, which are typically the most expensive aspect of a CFD process, the *eddy* solver is a general monolithic multiphyiscs solver.[28] The multigrid preconditioning approach has applicability to other governing equations used in multiphysics simulations. The current implementation leverages a uniform code base, and hence can be directly applied to any new physical system without modification.

References

[1]M. Ceze, L. Diosady, S. M. Murman, Development of a high-order space-time matrix-free adjoint solver, in: 54th AIAA Aerospace Sciences Meeting, 2016, p. 0833.

[2]C. Carton de Wiart, S. M. Murman, Assessment of wall-modeled les strategies within a discontinuous-galerkin spectral-element framework, in: 55th AIAA Aerospace Sciences Meeting, 2017, p. 1223.

[3]L. T. Diosady, S. M. Murman, Higher-order methods for compressible turbulent flows using entropy variables, in: 53rd AIAA Aerospace Sciences Meeting, 2015, p. 0294.

[4]A. Garai, L. T. Diosady, S. M. Murman, N. K. Madavan, Scale-resolving simulations of bypass transition in a high-pressure turbine cascade using a spectral element discontinuous galerkin method, Journal of Turbomachinery 140 (3) (2018) 031004.

[5]L. T. Diosady, S. M. Murman, Tensor-product preconditioners for higher-order space–time discontinuous galerkin methods, Journal of Computational Physics 330 (2017) 296–318.

[6]W. Pazner, P.-O. Persson, Approximate tensor-product preconditioners for very high order discontinuous galerkin methods, Journal of Computational Physics 354 (2018) 344–369.

[7]L. T. Diosady, S. M. Murman, Scalable tensor-product preconditioners for high-order finite-element methods: Scalar equations, Journal of Computational Physics.

[8]B. Helenbrook, D. Mavriplis, H. Atkins, Analysis of"p"-multigrid for continuous and discontinuous finite element discretizations, in: 16th AIAA Computational Fluid Dynamics Conference, 2003, p. 3989.

[9]F. Bassi, S. Rebay, Numerical solution of the euler equations with a multiorder discontinuous finite element method, in: Computational Fluid Dynamics 2002, Springer, 2003, pp. 199–204.

[10]K. J. Fidkowski, T. A. Oliver, J. Lu, D. L. Darmofal, p-multigrid solution of high-order discontinuous Galerkin discretizations of the compressible navier–stokes equations, Journal of Computational Physics 207 (1) (2005) 92–113.

[11]K. Shahbazi, D. J. Mavriplis, N. K. Burgess, Multigrid algorithms for high-order discontinuous Galerkin discretizations of the compressible navier–stokes equations, Journal of Computational Physics 228 (21) (2009) 7917–7940.

[12]L. T. Diosady, D. L. Darmofal, Preconditioning methods for discontinuous Galerkin solutions of the navier–stokes equations, Journal of Computational Physics 228 (11) (2009) 3917–3935.

[13]L. Botti, A. Colombo, F. Bassi, h-multigrid agglomeration based solution strategies for discontinuous Galerkin discretizations of incompressible flow problems, Journal of Computational Physics 347 (2017) 382–415.

[14]M. Franciolini, L. Botti, A. Colombo, A. Crivellini, p-multigrid matrix-free discontinuous galerkin solution strategies for the under-resolved simulation of incompressible turbulent flows, arXiv preprint arXiv:1809.00866.

[15]L. Botti, A. Colombo, A. Crivellini, M. Franciolini, {hp-hp}-multilevel discontinuous galerkin solution strategies for elliptic operators, International Journal of Computational Fluid Dynamics (2019) 1–9.

[16]A. Crivellini, M. Franciolini, A. Nigro, An implicit discontinuous galerkin method with reduced memory footprint for the simulation of turbulent flows, in: Direct and Large-Eddy Simulation XI, Springer, 2019, pp. 69–74.

[17]T. J. R. Hughes, L. Franca, M. Mallet, A new finite element formulation for computational fluid dynamics: I Symmetric forms of the compressible Euler and Navier-Stokes equations and the second law of thermodynamics, CMAME 54 (1986) 223–234.

[18]F. Ismail, P. L. Roe, Affordable, entropy-consistent euler flux functions ii: entropy production at shocks, J. Comput. Phys. 228 (15) (2009) 5410–5436.

[19]L. T. Diosady, S. M. Murman, General element shapes within a tensor-product higher-order space-time discontinuous-galerkin formulation, in: 22nd AIAA Computational Fluid Dynamics Conference, 2015, p. 3044.

[20]D. Knoll, D. Keyes, Jacobian-free Newton-Krylov methods: a survey of approaches and applications, J. Comput. Phys. 193 (2) (2004) 357–397.

[21]M. Franciolini, A. Crivellini, A. Nigro, On the efficiency of a matrix-free linearly implicit time integration strategy for high-order discontinuous Galerkin solutions of incompressible turbulent flows, Computers & Fluids 159 (2017) 276 – 294.

[22]A. Crivellini, M. Franciolini, A. Nigro, A matrix-free incompressible DG algorithm for the simulation of turbulent flows, in: Progress in Turbulence VII, Springer, 2017, pp. 153–159.

[23]P. F. Antonietti, M. Sarti, M. Verani, Multigrid algorithms for hp-discontinuous Galerkin discretizations of elliptic problems, SIAM Journal on Numerical Analysis 53 (1) (2015) 598–618.

[24]M. Franciolini, K. Fidkowski, A. Crivellini, Efficient discontinuous galerkin implementations and preconditioners for implicit unsteady compressible flow simulations, arXiv preprint arXiv:1812.04789.

[25]Y. Saad, A flexible inner-outer preconditioned gmres algorithm, SIAM Journal on Scientific Computing 14 (2) (1993) 461–469.

[26]D. Coles, A. J. Wadcock, Flying-hot-wire study of flow past an naca 4412 airfoil at maximum lift, AIAA Journal 17 (4) (1979) 321–329.

[27]S. R. Allmaras, F. T. Johnson, Modifications and clarifications for the implementation of the spalart-allmaras turbulence model, in: Seventh international conference on computational fluid dynamics (ICCFD7), 2012, pp. 1–11.

[28]C. Carton de Wiart, L. T. Diosady, A. Garai, N. Burgess, P. Blonigan, D. Ekelschot, S. M. Murman, Design of a modular monolithic implicit solver for multi-physics applications, in: 56th AIAA Aerospace Sciences Meeting, 2018, p. 1400.